Encapsulation

Le but de l'encapsulation

L'encapsulation est un concept primordial de l'orienté objet.

Il s'agit d'une règle consistant à cacher les données internes d'un objet, en autorisant l'accès à ces dernières uniquement avec des appels de méthodes.

L'encapsulation apporte de nombreux avantages :

- Elle permet un couplage faible des classes
- Elle permet de garantir l'intégrité des données internes d'une classe

Mise en place

Deux éléments nous permettent de mettre en place l'encapsulation :

- Les modificateurs de visibilité
- Les accesseurs et les mutateurs

Le langage python ne supportant pas les modificateurs de visibilité. Celui-ci utilise une convention d'écriture à l'aide de préfixe d'underscore, pour définir la visibilité "privé"...

- Un underscore : accessible directement via son nom.
- Deux underscore : accessible via un nom généré sous la forme *_Class__variable*

Si vous utilisez un IDE, un "Warning" est ajouté lorsque vous utilisez un élément "privé"

Les propriétés

Une propriété est un ensemble de méthodes qui permettent de contrôler les accès aux attributs de la classe.

Une propriété est composé de :

- Un accesseur (Getter) permet de consulter la valeur d'un champ
- Un mutateur (Setter) permet de modifier la valeur d'un champ

```python
class Voiture:

    def __init__(self):
        self.__roues = 4

    # Accesseur
    def _get_roues(self):
        return self.__roues

    # Mutateur
    def _set_roues(self, value):
        self.__roues = value

    # Definition de la propriété
    roues = property(_get_roues, _set_roues)


ma_voiture = Voiture()
ma_voiture.roues = 5
```

Syntaxe à l'aide des décorateurs

Le python dispose d'une syntaxe basé sur 2 décorateurs, qui permet de créer une propriété de manière simple.

Ceux-ci sont :

- *@property*
  Permet de définir la propriété, il se place sur l'accesseur

- *@<NomProp>.setter*
  Permet de définir le mutateur de la propriété.

```
class Voiture:

    def __init__(self):
        self.__roues = 4

    # Propriété & Accesseur
    @property
    def roues(self):
        return self.__roues

    # Mutateur
    @roues.setter
    def roues(self, value):
        self.__roues = value


ma_voiture = Voiture()
ma_voiture.roues = 5
print(ma_voiture.roues)
```